International Scientific Symposium «Intelligence Solutions 2021»

PERMUTATION-MATRIX APPROACH TO OPTIMAL ASSIGNMENT DESIGN

Olha Matsyi, Oksana Pichugina

Kharkiv National Automobile and Highway University,

National Aerospace University «Kharkiv Aviation Institute»

(Ukraine)

September 28-30, 2021

The paper presents new results on improving conventional algorithms for finding a perfect matchings of the minimum weight.

A new permutation-matrix model of optimal assignment in the balanced Linear Assignment Problem (LAP) is proposed, which allows recursively finding solutions on a set of augmenting paths built depending on the current matching.

Conventional methods for solving LAP, such as the Hungarian method, Kahn-Munkres method, and potential method, are built using different combinatorial optimization approaches. They are characterized by different time complexity, where the best known is $O\left(n^3\right)$, where n is the cost matrix order.

Our algorithm is designed to find a perfect matching of the minimum weight in a bipartite graph on $2n$ vertices.

Problem Statement

For the matrix of costs (weights) $C=\left[c_{ij}\right]_n$, where $C=\left[c_{ij}\right]_n$, where $c_{ij}\in R^1_{\geq 0}$ or $c_{ij}=+\infty$, find

$$C(\sigma)=\min_{\pi\in S_n} C(\pi), \text{ where } C(\pi)=\sum_{i=1}^{n} c_{i,\pi[i]}. \qquad (1)$$

Here S_n is a permutation group, $\pi=(\pi[1], \pi[2], ..., \pi[n])\in S_n$ represents a permutation from a set $J_n=\{1, 2,\ldots,n\}$ of the columns' indexes of the matrix C, $\sigma=(\sigma[1], \sigma[2], ..., \sigma[n])\in S_n$ is the optimal permutation yielding an optimal value of (1) $C(\sigma)=\sum_{i=1}^{n} c_{i,\sigma[i]}$. If $c_{i,\sigma[i]}\neq\infty$, $i=\overline{1, n}$, the LAP (1) is feasible and σ is a solution to the LAP.

Key Observation

Any part of a feasible solution of a LAP consisting of k elements uniquely determines a submatrix $\left[c_{i_s j_t}\right]_k$ of the matrix C, such that

$$i_1 < i_2 < \ldots < i_s < \ldots < i_k,\ j_1 < j_2 < \ldots < j_t < \ldots < j_k.$$

Introduce a sequence $\pi_k = \left(\pi_k[i_1],\ \pi_k[i_2],\ \ldots,\ \pi_k[i_s],\ \ldots, \pi_k[i_k]\right), \pi_k[i_s] \in \{\, j_1,\ j_2,\ \ldots,\ j_t,\ \ldots,\ j_k \,\}, k = \overline{1,\ n-1}$, such that: a) it is a solution of the LAP for the submatrix $\left[c_{i_s j_t}\right]_k$ as its cost matrix; b) the cost of the assignment associated with the permutation π_k does not exceed the optimal value of a LAP for any submatrix of the order k of the matrix C.

If there exists an efficient conversion procedure of a sequence π_k into a sequence π_{k+1}, $k = \overline{0,\ n-1}$, and the problem (1) has a solution, then it takes n steps to find $\sigma = \pi_n$.

Let us discuss how the sequences π_k, $k = \overline{1,\ n}$ can be constructed.

The Algorithm: Iteration 1

$k = 1$

The initial sequence $\pi_1 = (\pi_1[i_1])$ is trivially defined.

Namely, the matrix C contains

$$c_{lr} = \min\{c_{ij} \mid 1 \le i, j \le n\},$$

therefore, we set

$$i_1 = l, \pi_1[i_1] = r.$$

The Algorithm: Iteration 2

$k = 2$. In order to find $\pi_2 = (\pi_2[i_1], \pi_2[i_2])$ based on π_1 let us determine

$$c_{ms} = \min\{c_{ij} \mid i \neq l,\ j \neq r\},\ c_{lp} = \min\{c_{lj} \mid j \neq r\},\ c_{vr} = \min\{c_{ir} \mid i \neq l\}$$ (see Fig. 1).

		p	q		r		s	
m		c_{mp}					c_{ms}	
l		c_{lp}			c_{lr}			
v					c_{vr}		c_{vs}	
w			c_{wq}					

Figure 1.

The Algorithm: Iteration 2 (cont.)

It is easy to see that if $c_{lr} + c_{ms} \le c_{lp} + c_{vr}$ (further Case 1) then conditions a) and b) are satisfied for a sequence π_2, where $i_1 = l,\ \pi_2[i_1] = r,\ i_2 = m,\ \pi_2[i_2] = s$.

It corresponds to a submatrix of A shown in Fig. 2.

$$\begin{array}{c|c|c|} & r & s \\ \hline l & c_{lr} & \\ \hline m & & c_{ms} \\ \hline \end{array}$$

Figure 2

Otherwise, i.e. $c_{lr} + c_{ms} > c_{lp} + c_{vr}$ (further Case 2), these conditions are satisfied on a sequence π_2 with elements $i_1 = l,\ \pi_2[i_1] = p,\ i_2 = v,\ \pi_2[i_2] = r$ and the corresponding submatrix is depicted in Fig. 3.

$$\begin{array}{c|c|c|} & p & r \\ \hline l & c_{lp} & \\ \hline v & & c_{vr} \\ \hline \end{array}$$

Figure 3

The Algorithm: Iteration 3

$k=3$. Now, we transform the sequence π_2 into a sequence $\pi_3=\left(\pi_3[i_1],\ \pi_3[i_2],\ \pi_3[i_3]\right)$.

Without loss of generality, we can assume that we deal with Case1, i.e. $\pi_2=\left(\pi_2[l]=r,\ \pi_2[m]=s\right)$.

Then we find $c_{wq}=\min\left\{c_{ij}\mid i\neq l,\ m;\ \ j\neq s,\ r\right\}$ and

$$MIN1=c_{lr}+c_{ms}+c_{wq}. \qquad (2)$$

Note that $c_{wq}=c_{ms}$, if we deal with Case 2, i.e. $\pi_2=\left(\pi_2[l]=p,\ \pi_2[v]=r\right)$.

The transformation of π_2 into π_3 is the result of solving the following auxiliary problem. For rows $l,\ m$ and columns $r,\ s$ of the matrix C, it is required to find a triple of components with the minimum sum of their values. Any two elements of the triple must be placed in three different rows and three various columns including $r,\ s$.

The Algorithm: Iteration 3 (cont.)

If such a triple does not contain c_{lr}, but includes c_{ms} (further **Case 1'**), then the sum of its elements is bounded from below by the value

$S_1 = c_{vr} + c_{lp} + c_{ms}$, where $c_{vr} = \min\{c_{ir} \mid i \neq l, m\}$, $c_{lp} = \min\{c_{lj} \mid j \neq r, s\}$.

Let the solution of the auxiliary problem be a triple, which includes c_{lr} and does not contain c_{ms} (further **Case 2'**). Then a value $S_2 = c_{lr} + c_{mp} + c_{vs}$, where $c_{mp} = \min\{c_{mj} \mid j \neq s, r\}$, $c_{vs} = \min\{c_{is} \mid i \neq l, m\}$.

Case 3'. Elements of a solution of the auxiliary problem that does not contain c_{lr} and c_{ms} define a value , where

$$c_{ws} = \min\{c_{is} \mid i \neq l, m\},\ c_{mr} = \min\{c_{mj} \mid j \neq p, s\},\ c_{ls} = \min\{c_{is} \mid i \neq m, v\},$$

$$c_{mq} = \min\{c_{mj} \mid j \neq r, s\} \text{ (see Fig. 4).}$$

The Algorithm: Iteration 3 (cont.)

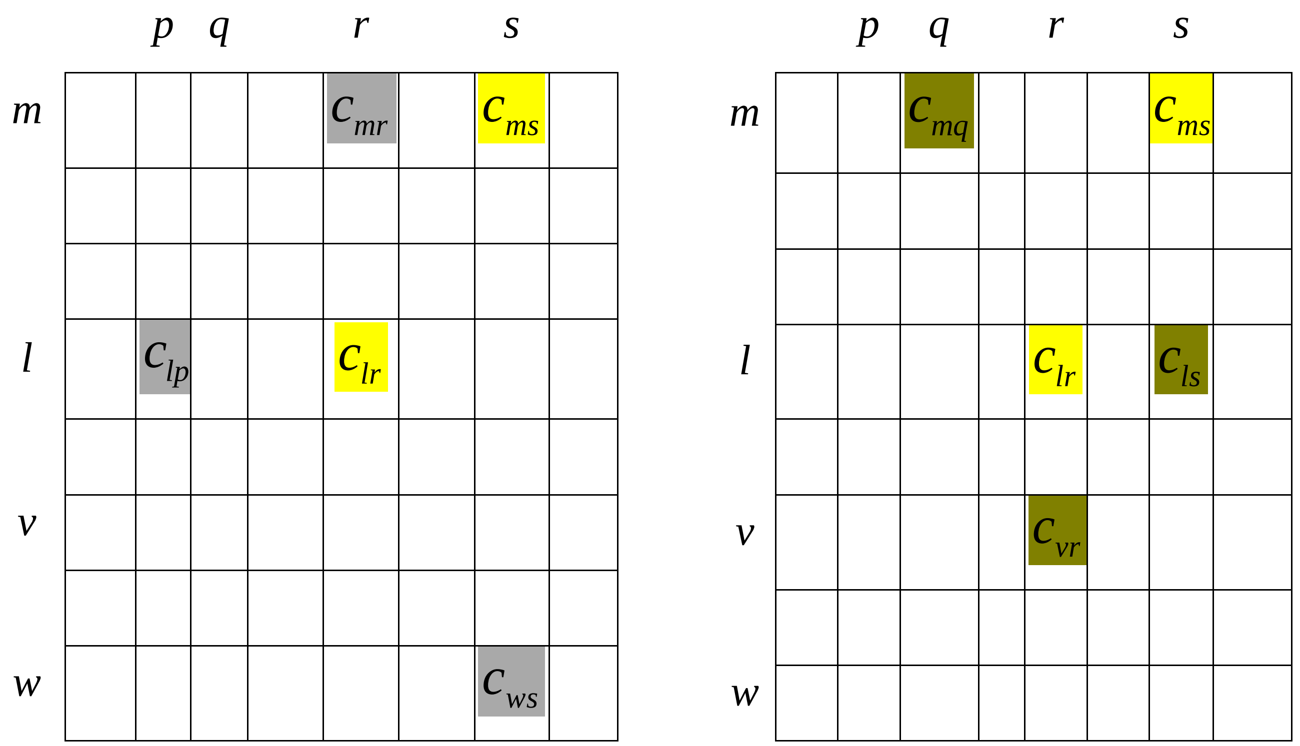

Figure 4 (left - S_3-component 1, right - S_3-component 2)

The Algorithm: Iteration 3 (cont.)

Let $MIN2 = \min\{S_1, S_2, S_3\}$. It is clear that it corresponds to the sequence π_3 (see the indices in Cases 1'-3'), if $MIN2 < MIN1$, otherwise $\pi_3 = (\pi_3[l] = r,\ \pi_3[m] = s,\ \pi_3[w] = q)$ (see (2)).

$k = 3 \to k = \overline{1,n}$ In general, a sequence $\pi_k = (\pi_k[i_1], \pi_k[i_2], ..., \pi_k[i_s], ..., \pi_k[i_k])$ having properties a) and b) is transformed into a sequence $\pi_{k+1} = (\pi_{k+1}[i_1],\ \pi_{k+1}[i_2], ..., \pi_{k+1}[i_r], ..., \pi_{k+1}[i_{k+1}])$ with the same properties as the following.

If, in the matrix C, it is found

$$c_{\pi_k[i_{k+1}]} = \min\{c_{ij} \mid i \neq i_1, i_2, ..., i_s, ..., i_k,\ j \neq \pi_k[i_1], \pi_k[i_2], ..., \pi_k[i_s], ..., \pi_k[i_k]\},$$

then a sequence $\pi_{k+1}^1 = (\pi_k, \pi_k[i_{k+1}])$ is , and it is evaluated

$$MIN1 = \sum_{s=1}^{k} c_{\pi_k[i_s]} + c_{\pi_k[i_{k+1}]}.$$

The Algorithm: Iteration $k+1$

Next, the problem of finding $k+1$ of C is solved, for which in the matrix the minimum sum of their values is attained while they are located in different rows and columns and include all rows and columns with numbers specified by values $c_{\pi_k[i_1]}$, $c_{\pi_k[i_2]}$, ..., $c_{\pi_k[i_s]}$, ..., $c_{\pi_k[i_k]}$.

In the same manner, let us denote this sum by $MIN2$.

The found elements form the required sequence $\pi_{k+1} = \pi^2_{k+1}$, if $MIN2 \le MIN1$.

Otherwise, $\pi_{k+1} = \pi^1_{k+1}$.

CONCLUSION

A new permutation-matrix model of optimal assignment is proposed. It allows recursively finding solutions on a set of augmenting paths built with respect to the current matching. The proposed scheme for finding an optimal assignment underlies a method of solving a LAP, where a solution to the problem is found by means of matching theory for bipartite graphs only.

The offered approach for finding the optimal assignment develops transport logistics' theory. It is focused on improving the organization of transportation in real-time and in real situations of vehicle traffic. Its implementing allows reducing the time and fuel consumption for carrying out transportations.

The method uses the well-known algorithm for finding a perfect matching in bipartite unweighted graphs, which is built according to a scheme that expands approaches to solving complex optimization problems.

THANK YOU!